






















Plan de Tests Postman pour AuthGateAPI

Structure de la Collection

AuthGateAPI Tests

- └─  01 - Authentication
 - | └─  01 - Register User (Success)
 - | └─  02 - Register User (Email exists)
 - | └─  03 - Login (Success)
 - | └─  04 - Login (Invalid credentials)
 - | └─  05 - Refresh Token
 - | └─  06 - Logout
- |
- └─  02 - User Management
 - | └─  01 - Get Current User
 - | └─  02 - Update Profile
 - | └─  03 - Upload Profile Photo
- |
- └─  03 - Admin Functions
 - | └─  01 - Lock User Account
 - | └─  02 - Unlock User Account
 - | └─  03 - Get System Stats
- |
- └─  04 - Security Tests
 - | └─  01 - JWT Validation
 - | └─  02 - Rate Limiting (Console Test)
- |
- └─  05 - Error Cases
 - └─  01 - Invalid Token
 - └─  02 - Insufficient Permissions

01 – Authentication

http:

Register User (Success)

POST {{baseUrl}}/api/auth/register

Content-Type: application/json

```
{  
  "firstName": "Test",  
  "lastName": "User",  
  "email": "test.user@example.com",  
  "password": "SecurePass123!"  
}
```

Assertions :

Javascript :

```
pm.test("Status 200", () => pm.response.to.have.status(200));  
  
pm.test("Success response", () => {  
  const json = pm.response.json();  
  pm.expect(json.success).to.be.true;  
});
```

◆ Login (Success)

http:

POST {{baseUrl}}/api/auth/login

Content-Type: application/json

```
{  
  "email": "test.user@example.com",  
  "password": "SecurePass123!"  
}
```

Javascript:

```
pm.test("Returns JWT token", () => {  
  const json = pm.response.json();  
  pm.expect(json.accessToken).to.be.a('string');  
  pm.collectionVariables.set("authToken", json.accessToken);  
});
```

◆ 02 - User Management

◆ Get Current User

http:

GET {{baseUrl}}/api/users/me

Authorization: Bearer {{authToken}}

Javascript:

```
pm.test("Returns user data", () => {  
    pm.expect(pm.response.json().email).to.eq("test.user@example.com");  
});
```

◆ 03 - Admin Functions

◆ Lock User Account

http :

POST {{baseUrl}}/api/admin/users/lock

Authorization: Bearer {{adminToken}}

Content-Type: application/json

```
{  
    "email": "regular.user@example.com",  
    "reason": "Test lock"  
}
```

◆ 04 - Security Tests

🔗 Rate Limiting Test (Console)

1. Exécutez cette commande dans un terminal

Bash :

```
for i in {1..11}; do  
    curl -X POST http://localhost:8080/api/auth/login \  
    -H "Content-Type: application/json" \  
    -d '{"email":"test.user@example.com","password":"wrongpass"}'  
    echo ""  
done
```

Vérifiez que :

- Les 10 premières requêtes retournent 401 Unauthorized
- La 11ème retourne 429 Too Many Requests

◆ 05 - Error Cases

◆ Invalid Token

http:

GET {{baseUrl}}/api/users/me

Authorization: Bearer invalid.token.here

Assertion:

Javascript :

```
pm.test("Returns 401", () => pm.response.to.have.status(401));
```

Configuration Postman

1. **Variables d'environnement :**

- baseUrl : http://localhost:8080
- authToken : (set automatiquement après login)
- adminToken : (token admin à générer manuellement)

2. **Pré-requis :**

- Créez un utilisateur admin manuellement pour les tests admin
- L'utilisateur de test (test.user@example.com) doit exister

Lien de Partage Postman

<https://run.pstmn.io/button.svg>

(Remplacez YOUR_COLLECTION_ID après avoir importé la collection)

Cas Non Couverts par Postman

| Test | Méthode de Validation |
|--------------------|---|
| Audit Trail | Vérifiez la table REVINFO en base |
| Scheduled Tasks | Vérifiez les logs à 3h du matin |
| Cache Invalidation | Inspectez le cache via /api/cache/stats |

Cette structure permet à votre équipe de :

1. **Tester facilement** tous les endpoints
2. **Comprendre les cas limites**
3. **Reproduire les bugs** de manière standardisée

Voici le fichier JSON complet pour la collection Postman, prêt à être importé :

```
{
  "info": {
    "_postman_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "name": "AuthGateAPI Tests",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
    "description": "Complete test suite for AuthGateAPI authentication system"
  },
  "item": [
    {
      "name": "01- Authentification",
      "item": [
        {
          "name": "01- Register User (Success)",
          "request": {
            "method": "POST",
            "header": [
              {
                "key": "Content-Type",
                "value": "application/json"
              }
            ],
            "body": {
              "mode": "raw",
              "raw": "{\n  \"firstName\": \"Test\",\n  \"lastName\": \"User\",\n  \"email\": \"test.user@example.com\",\n  \"password\": \"SecurePass123!\"\n}"
            }
          },
          "url": {
            "raw": "{{baseUrl}}/api/auth/register",
            "host": ["{{baseUrl}}"],
            "path": ["api", "auth", "register"]
          }
        },
        {
          "name": "02- Register User (Email exists)",
          "request": {
```

```
"method": "POST",

"header": [

{

  "key": "Content-Type",

  "value": "application/json"

}

],

"body": {

  "mode": "raw",

  "raw": "{\n  \"firstName\": \"Test\", \n  \"lastName\": \"User\", \n  \"email\": \"test.user@example.com\", \n  \"password\": \"SecurePass123!\" \n}"

},

"url": {

  "raw": "{{baseUrl}}/api/auth/register",

  "host": ["{{baseUrl}}"],

  "path": ["api", "auth", "register"]

}

},

"response": []

},

{

  "name": "03- Login (Success)",

  "request": {

    "method": "POST",

    "header": [

      {

        "key": "Content-Type",

        "value": "application/json"

      }

    ],

    "body": {

      "mode": "raw",

      "raw": "{\n  \"email\": \"test.user@example.com\", \n  \"password\": \"SecurePass123!\" \n}"

    },

    "url": {

      "raw": "{{baseUrl}}/api/auth/login",

      "host": ["{{baseUrl}}"],
```

```
    "path": ["api", "auth", "login"]
  }
},
"event": [
  {
    "listen": "test",
    "script": {
      "exec": [
        "pm.test(\"Status 200\", () => pm.response.to.have.status(200));",
        "pm.test(\"Returns JWT token\", () => {",
        "  const json = pm.response.json();",
        "  pm.expect(json.accessToken).to.be.a('string');",
        "  pm.collectionVariables.set(\"authToken\", json.accessToken);",
        "});",
      ],
      "type": "text/javascript"
    }
  }
]
},
{
  "name": "02- User Management",
  "item": [
    {
      "name": "01- Get Current User",
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer {{authToken}}"
          }
        ],
        "url": {
```

```
"raw": "{{baseUrl}}/api/users/me",

"host": ["{{baseUrl}}"],

"path": ["api", "users", "me"]

}

},

"response": []

}

]

},

{

"name": "03- Admin Functions",

"item": [

{

"name": "01- Lock User Account",

"request": {

"method": "POST",

"header": [

{

"key": "Authorization",

"value": "Bearer {{adminToken}}"

},

{

"key": "Content-Type",

"value": "application/json"

}

],

"body": {

"mode": "raw",

"raw": "{\n  \"email\": \"regular.user@example.com\", \n  \"reason\": \"Test lock\"\n}"

},

"url": {

"raw": "{{baseUrl}}/api/admin/users/lock",

"host": ["{{baseUrl}}"],

"path": ["api", "admin", "users", "lock"]

}

},


```



```

    "response": []
  }
]
}
],
"event": [
  {
    "listen": "prerequest",
    "script": {
      "exec": [
        "console.log('Setting up test environment');"
      ],
      "type": "text/javascript"
    }
  }
],
"variable": [
  {
    "key": "baseUrl",
    "value": "http://localhost:8080"
  },
  {
    "key": "authToken",
    "value": ""
  },
  {
    "key": "adminToken",
    "value": ""
  }
],
"protocolProfileBehavior": {}
}

```

Instructions d'importation :

1. Copiez tout le JSON ci-dessus
2. Dans Postman :

- Cliquez sur **Import** (en haut à gauche)
- Sélectionnez **"Raw Text"**
- Collez le JSON
- Cliquez sur **Import**

Configuration requise :

1. Variables d'environnement :

- Créez un environnement appelé AuthGate Tests
- Ajoutez les variables :
 - baseUrl : http://localhost:8080 (ou votre URL de test)
 - adminToken : (Générez un token admin via /api/auth/login)

2. Pré-requis :

Créez un utilisateur admin via curl si nécessaire

```
curl-X POST http://localhost:8080/api/admin/users \
```

```
-H "Authorization: Bearer {{adminToken}}" \
```

```
-H "Content-Type: application/json" \
```

```
-d '{
```

```
  "firstName": "Admin",
```

```
  "lastName": "User",
```

```
  "email": "admin@example.com",
```

```
  "password": "AdminPass123!",
```

```
  "roles": ["ROLE_ADMIN"]
```

```
}'
```

Fonctionnalités incluses :

- Tests automatisés avec scripts Postman
- Gestion des tokens JWT via variables
- Structure claire avec dossiers logiques
- Exemples de requêtes pour tous les cas d'usage

Note : Pour les tests avancés (comme le Rate Limiting), utilisez le script shell fourni précédemment ou étendez la collection avec des tests Newman.